An Exploratory Study on Process Representations

A Thesis Presented

by

Chetan Nagaraj Naik

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

Stony Brook University

August 2016

Stony Brook University

The Graduate School

Chetan Nagaraj Naik

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis

Dr. Niranjan Balasubramanian – Thesis Advisor
Research Assistant Professor, Department of Computer Science

Dr. H. Andrew Schwartz – Chairperson of Defense
Assistant Professor, Department of Computer Science

Dr. Dani Yogatama
Research Scientist, Google DeepMind

This thesis is accepted by the Graduate School.

Nancy Goroff
Interim Dean of the Graduate School

Abstract of the Thesis

An Exploratory Study on Process Representations

by

Chetan Nagaraj Naik

Master of Science

in

Computer Science

Stony Brook University

2016

Knowledge about processes is essential for AI systems to understand and reason about the real world events. And, the systems need some form of semantic representation to perform reasoning. At the simplest level, even knowing which class of entities play key roles can be helpful in recognizing and reasoning about events. For instance, given a description “a puddle drying in the sun", one can recognize this as an instance of the evaporation process using simple role knowledge which asserts (among other things) that the undergoer is a kind of liquid (the puddle), and the enabler is a heat source (the sun).

In this work, we explore two forms of process knowledge representations, a frame representation with a fixed set of roles and a matrix representation. We developed a fully feature engineered and a non engineered (using deep LSTM) system for role classification. We improve these process knowledge extraction models by performing *cross-sentence* inference—over role classifier scores—which extends the standard *within sentence* joint inference to inference across multiple sentences. We also present our preliminary work on modeling processes as operators.

Dedicated to my parents.

Contents

List of Figures

List of Tables

Acknowledgements

The last two years at Stony Brook University have been the best years of my life till now. It has been an immense learning experience, in more ways than one. I wish to thank everyone who has been part of this journey.

I have been extremely lucky to have as my advisor, Prof. Niranjan Balasubramanian, who provided me with a perfect balance of guidance and freedom. He made sure that I am never stressed out by letting me take holidays as needed. He also supported me immensely while I was going through tough times in life. I am also thankful to Dani Yogatama for many valuable technical discussions and helpful career advice.

I would like to thank all members of NLP lab, and especially Samuel Louvan, with whom I worked more closely. I would like to thank my friends Nelson George, Calvin Rego, Shashank Shanbhag, Darshan Hegde, Vinaykumar Hegde, Ankit Arun, Manoj Alwani, Rakesh Chada, Rumit Desai, Nitish Gupta and everyone else who encouraged, challenged and motivated me throughout.

I wouldn't be where I am today without the amazing support, encouragement and love from my parents Nagaraj KT and Geetha and my brother Chandan. They taught me the importance of education in life, and always supported me in my endeavours.

Chapter 1

Introduction

1.1 Overview

One of the goals of Artificial Intelligence (AI) is to understand and reason about the real world events and scenarios. To achieve this, it is necessary for the AI systems to learn to identify and disentangle the underlying explanatory factors hidden in the data. Hence the success of AI systems generally depends on data representations [3].

Many of the real world events and scenarios are expressed in natural language texts. Understanding the events from text requires Natural Language Processing (NLP). At a high level, understanding an event means being able to answer questions about it. Numerous practical applications could take advantage of language understanding, for example, to extract actionable knowledge, answer questions, or summarize events based on the semantic content of a text. And understanding events/concepts need some form of semantic representation which provides an abstraction from lexical and syntactic realizations. With better natural language semantic representations, computers can do a better job of answering questions as a result of better understanding of natural text.

In this work, we focus on exploring two forms of representations to store knowledge about processes (biological, chemical, physical) in order to answer 4th grade level questions.

1. FRAME REPRESENTATION: At the 4th grade level, the questions do not involve deep knowledge about the sub-events or their sequential order. Rather the questions test for shallower knowledge about the entities undergoing change, the resulting artifacts, and the main characteristic action describing the process. This knowledge is naturally expressed via semantic roles. Accordingly we design a simple representation that encodes information about each process via the following roles [4]:

   (a) *Input* – This role captures the main input to the process or the object undergoing the process. *e.g.,* Water is an input to the evaporation process.

   (b) *Result* – The artifact that results from the process or the change that results from the process *e.g.,* Water vapor is a result of evaporation.

   (c) *Trigger* – The main action, expressed as a verb or its nominalization, indicating the occurrence of the process. *e.g.,* converted is a trigger for evaporation.

   (d) *Enabler* – The artifact, condition or action that enables the process to happen. *e.g.,* Sun is a heat source that is enabler for evaporation.

2. MATRIX REPRESENTATION: We encode the process knowledge in a matrix that acts as an operator which predicts the output of a process given its input.

This thesis also describes methods to extract process knowledge—that conforms to the above mentioned representations—from natural language text.

1. **Process Knowledge Extraction using Feature Engineering**: Here we describe a system that uses feature engineered semantic role label-

ing (SRL) system and Integer Linear Program (ILP) to extract process knowledge (FRAME REPRESENTATION).

2. **Process Knowledge Extraction using LSTM**: Here we describe a system that uses a combination of Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) and ILP to extract process knowledge (FRAME REPRESENTATION).

3. **Learning Process Operators**: Here we learn a process operator matrix that maps process input to a feature space where both inputs and outputs of processes are represented.

1.2 Thesis Outline

Chapter 2 briefly describes Semantic Role Labeling and Recurrent Neural Networks in order to lay down the background of our work.

Chapter 3 describes our system for Cross-Sentence Inference using engineered features with ILP and its extensions.

Chapter 4 describes our system for process knowledge extraction using LSTM and cross-sentence inference using ILP.

Chapter 5 describes our method to model processes as operators.

In Chapter 6 we conclude by outlining future work and further research directions.

Chapter 3.1 is a joint work with Samuel Louvan which has been selected for publication at EMNLP 2016 [5] and hence appear as is (except for minor modifications to suit thesis requirements).

Chapter 2

Background

This chapter provides the background material and literature review for semantic role labeling and recurrent neural networks. Section 2.1 briefly reviews semantic role labeling in general. Section 2.2 reviews deep learning and related sequence–to–sequence methods. Section 2.3 covers related work in representation and extraction of semantic knowledge.

2.1 Semantic Role Labeling

As discussed earlier, to understand events we need some form of semantic representation which provides an abstraction from lexical and syntactic realizations. In NLP literature, events are often referred to as predicates and the participants attached to the predicates as its arguments. A predicate and its arguments form a predicate-argument structure. Semantic Role Labeling (SRL) [6] is a task that involves prediction of predicate-argument structure, *i.e.,* both identification of arguments as well as assignment of labels according to their underlying *semantic role*. SRL representations have many potential applications in NLP and have been shown to benefit question answering [7, 8], textual entailment [9], machine translation [10–12], and dialogue systems [13, 14], among others.

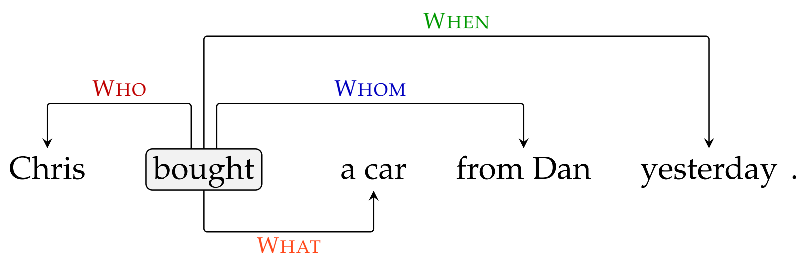

Figure 2.1: Predicate-argument structure in SRL

Consider Figure 2.1, the processing of the sentence should result in identifying the predicate *bought* involving *Chris* as the Agent, *Dan* as the Seller, *car* as the item being bought and *yesterday* as the time when the event happened. Computational systems can make use of these semantic roles as shallow semantic representation to make inferences that is not possible by only using surface words or syntactic representation. For example, it can be used to answer queries such as:

Who bought a car from Dan?
From *whom* did Chris buy a car ?
What did Chris buy?
When did Chris buy a car from Dan?

SRL is a challenging task because the same event can be expressed in several varying syntactic forms. The previous example can be stated in different ways: A car was bought by Chris, A car was sold to Chris, Chris was sold a car by Dan, etc.

Different frameworks for representing semantic predicate-argument structure have been established, notably FrameNet, VerbNet and PropBank, with accompanying sense and role inventories and annotated resources.

- **FrameNet:** offers a full-fledged semantic predicate-argument representation. Predicates trigger a prototypical situation, called frame, that defines the possible participants in the situation and their semantic roles in relation to that predicate.

- **PropBank:** provides a small role inventory. Labels lack semantic transparency and are marked as A0 to A5. Adjuncts are tagged with a small set of labels, such as ArgM-LOC.

- **VerbNet:** is located between FrameNet and PropBank on a continous scale between a fine-grained interpretable role inventory on one side and a compact, coarse-grained inventory on the other.

2.2 Deep Learning

Most of the current machine learning methods use human-designed feature representations. And, the task of the machine learning method becomes merely about optimizing feature weights needed to make the best final prediction. Deep Learning is a method that combines representation learning with machine learning. It allows computational models to jointly learn representations of data with multiple levels of abstraction, and the final prediction. Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity [15, 16].

Deep learning theory shows that deep nets have two different exponential advantages over classic learning algorithms that do not use distributed representations [17]. Both of these advantages arise from the power of composition and depend on the underlying data-generating distribution having an appropriate componential structure [3]. First, learning distributed representations enable generalization to new combinations of the values of learned features beyond those seen during training (for example, 2^n combinations are possible with n binary features) [18]. Second, composing layers of representation in a deep net brings the potential for another exponential advantage (exponential in the depth) [19].

2.2.1 Recurrent Neural Networks

Recurrent neural networks (RNN) [20] are a family of neural networks for processing sequential data. RNNs can scale to much longer sequences than would be practical for networks without sequence-based specialization. They can also process sequences of variable length.

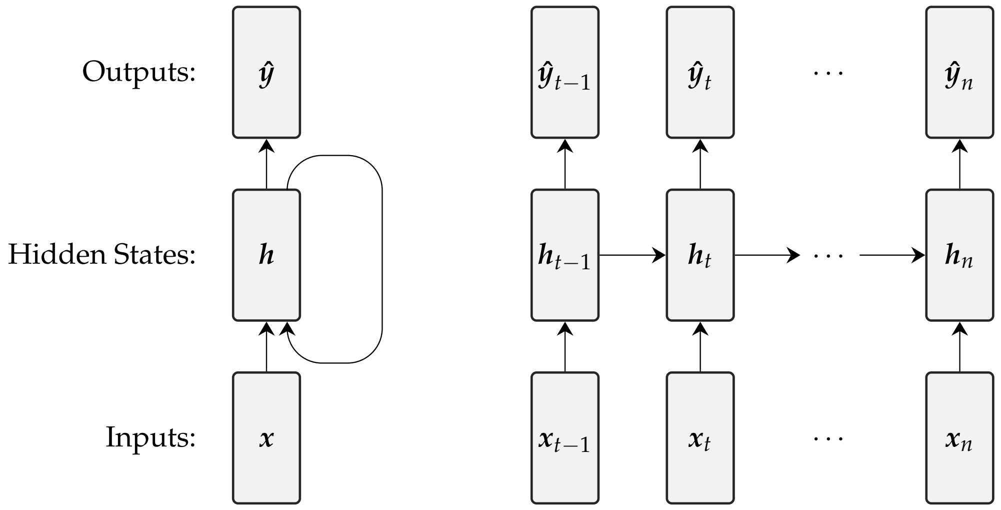

Figure 2.2: A recurrent neural network and the unfolding in time of the computation involved in its forward computation.

RNNs process an input sequence one element at a time, maintaining in their hidden units a 'state vector' that implicitly contains information about the history of all the past elements of the sequence. RNN structure is illustrated in Figure 2.2. In this figure, we mark input layer as $\boldsymbol{x}$, hidden layer as $\boldsymbol{h}$ and output layer as $\hat{\boldsymbol{y}}$. When considering the structure of RNN, people usually unfold it in time to deal with it more easily. The basic idea of unfolding RNN in time is to copy RNN several times and connect them in a chronological order. In Figure 2.2, the right structure illustrates the unfolded version of the network in the left. When we consider the outputs of the hidden units at different discrete (unfolded) time steps as if they were the outputs of different neurons in a deep multilayer network, it becomes clear how we can apply backpropagation to train RNNs.

Many recurrent neural networks use equation 2.1 or a similar equation to define the values of their hidden units.

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \theta) \tag{2.1}$$

where θ represents the parameters of the network. The same parameters (the same value of θ used to parametrize f) are used for all time steps.

When the recurrent network is trained to perform a task that requires predicting the future from the past, the network typically learns to use h_t as a kind of lossy summary of the task-relevant aspects of the past sequence of inputs up to t. This summary is in general necessarily lossy, since it maps an arbitrary length sequence $(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$ to a fixed length vector h_t. Depending on the training criterion, this summary might selectively keep some aspects of the past sequence with more precision than other aspects.

For $t = 1$ to $t = \tau$ *(number of time steps)*, we apply the following update equations:

$$\boldsymbol{a}_t = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t \tag{2.2}$$

$$\boldsymbol{h}_t = \tanh(\boldsymbol{a}_t) \tag{2.3}$$

$$\boldsymbol{o}_t = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}_t \tag{2.4}$$

$$\hat{\boldsymbol{y}}_t = \text{softmax}(\boldsymbol{o}_t) \tag{2.5}$$

where the parameters are the bias vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ along with the weight matrices $\boldsymbol{U}, \boldsymbol{V}$ and $\boldsymbol{W}$, respectively for input-to-hidden, hidden-to-output and hidden-to-hidden connections. This is an example of a recurrent network that maps an input sequence to an output sequence of the same length. The total loss for a given sequence of $\boldsymbol{x}$ values paired with a sequence of $\boldsymbol{y}$ values would then be just the sum of the losses over all the time steps. [15]

The back-propagation algorithm applied to the unrolled RNN is called **back-propagation through time** or BPTT. Though RNNs are very powerful dynamic systems, training them has proved to be problematic because the back-propagated gradients either grow or shrink at each time step, so over many time steps they typically explode or vanish. To prevent this forgetting problem, various techniques have been developed. *Structurally Constrained Recurrent Nets* (SCRN) [21] split the hidden state into fast and slow changing parts to solve the problem. Other set of solutions like *Long Short-Term Memory* (LSTM) [22] and *Gated Recurrent Units* (GRU) [23] use gating units for internal states to solve the issue.

2.2.2 Long Short-Term Memory (LSTM)

As discussed, an important benefit of recurrent networks is their ability to use contextual information when mapping between input and output sequences. Unfortunately, for standard RNN architectures, the range of context that can be accessed is limited because of the exploding and vanishing gradient problem [24–26]. In practice this shortcoming makes it hard for an RNN to learn tasks containing delays of more than about 10 time-steps between relevant input and target events [25]. The most effective solution so far is the Long Short-Term Memory (LSTM) architecture [22].

The LSTM architecture consists of a set of recurrently connected subnets, known as memory blocks. These blocks can be thought of as a differentiable version of the memory chips in a digital computer. Each block contains one or more self-connected memory cells and three multiplicative units — the input, output and forget gates — that provide continuous analogues of write, read and reset operations for the cells.

Figure 2.3 provides an illustration of an LSTM memory block with a single cell. An LSTM network is formed exactly like a simple RNN, except that the nonlinear units in the hidden layer are replaced by memory blocks. The LSTM memory cell stores information of the sequential data. The multiplicative gates remember when and how much the information in memory cell should be updated. This allows LSTM to store and access information over long periods of time, thereby avoiding the vanishing gradient problem. Mathematically, the process is defined by Formulas from 2.6 to 2.10.

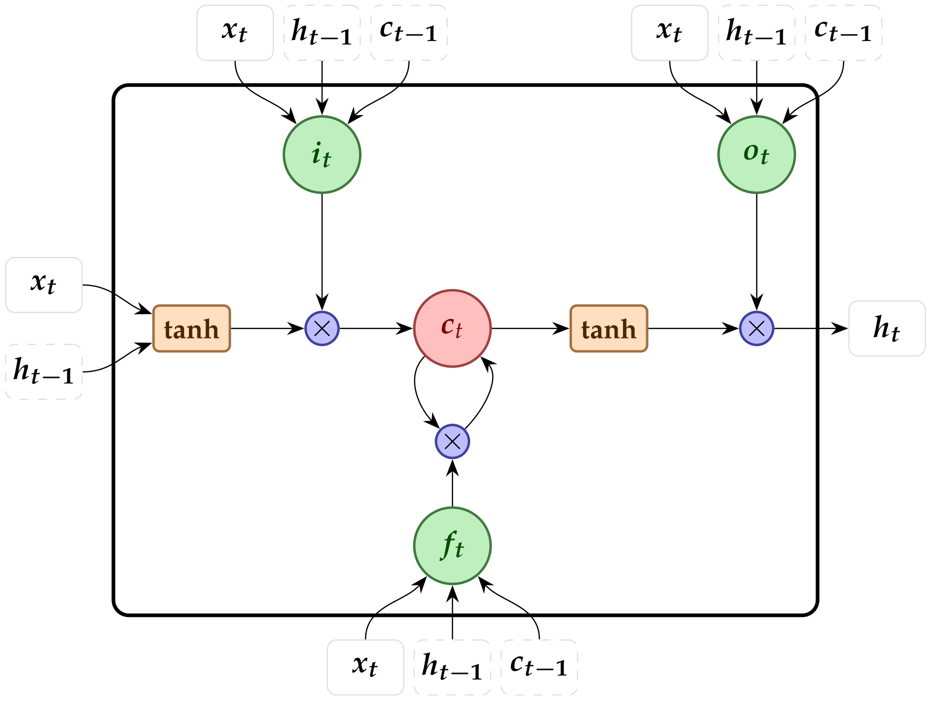

Figure 2.3: **LSTM memory block with one cell.** In the middle, there is the memory cell c_t which keeps the information of the data sequence. Around it, there are three gates, namely input gate i_t, output gate o_t and forget gate f_t [1, 2]. Each of them gets information from the input and controls the updating rule of memory cell.

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \tag{2.6}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \tag{2.7}$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{2.8}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \tag{2.9}$$

$$h_t = o_t \tanh(c_t) \tag{2.10}$$

where σ is the logistic sigmoid function, and i, f, o and c are respectively the *input gate*, *forget gate*, *output gate* and *cell input* activation vectors, all of which are the same size as the hidden vector h. The weight matrix subscripts have the obvious meaning, for example W_{hi} is the hidden-input gate matrix, W_{xo}

is the input-output gate matrix etc. The weight matrices from the cell to gate vectors (e.g. W_{ci}) are diagonal, so element m in each gate vector only receives input from element m of the cell vector. [27]

2.3 Related Work

Role-based representations have been shown to be useful for open-domain factoid question answering [7, 28], and comprehension questions on process descriptions [29]. Similar to process comprehension work, we target semantic representations about processes, but we focus only on a high-level summary of the process, rather than a detailed sequential representation of sub-events involved. Moreover, we seek to aggregate knowledge from multiple descriptions rather than understand a single discourse about each process.

There has been substantial prior work on semantic role labeling itself, that we leverage in this work. First, there are several systems trained on the PropBank dataset, e.g., EasySRL [30], Mate [31], Generalized-Inference [32]. Although useful, the PropBank roles are verb (predicate) specific, and thus do not produce consistent labels for a *process* (that may be expressed using several different verbs). In contrast, frame-semantic parsers, e.g., SEMAFOR [33], trained on FrameNet-annotated data [34] do produce concept (frame)-specific labels, but the FrameNet training data has poor ($< 50\%$) coverage of process verbs. Building a resource like FrameNet for a list of scientific processes is expensive.

Several unsupervised, and semi-supervised approaches have been proposed to address these issues for PropBank style predicate-specific roles [35–40]. A key idea here is to cluster syntactic signatures of the arguments and use the discovered clusters as roles. Another line of research has sought to perform joint training for syntactic parsing and semantic role labeling [30], and in using PropBank role labels to improve FrameNet processing using pivot features [41].

Our goal is to acquire a high quality semantic role based knowledge about processes. This allows us a unique opportunity to jointly interpret sentences that are discussing the same process. We build on ideas from previous within sentence joint inference [32], argument similarity notions in semi and unsupervised approaches [38], and combining PropBank roles to propose a cross-sentence inference technique [41]. The inference can be integrated with existing trained supervised learning pipelines, which can provide a score for role assignments for a given span.

Recently, there has been increased interest in using deep learning methods for a variety of language and information retrieval applications. In [42] Collobert et al. proposed models that can perform many NLP tasks with very little feature engineering. Unfortunately, the model restricts the use of context to a fixed size window around each word, this makes it hard for the model to learn long-distance relation between the words. But, this approach inspired Zhou and Xu [43], who solved the issue using deep bidirectional LSTMs and achieved state-of-the-art results.

Processes are complex events and hence can be seen as functions acting on a number of arguments rather than simple lexical units with associated word vectors. In the simplest setting, verbs and adjectives can also be considered as functions rather than simple lexical units. Baroni and Zamparelli [44], modeled adjectives as matrices which were estimated with partial least squares regression. The adjective matrix multiplied with a vectorial representation for noun will produce a vectorial representation of the specific adjective-noun compound. Socher et al. [45] proposed a model that assigns each word a vector and a matrix. The words are composed via a nonlinear function to create phrase representations consisting of another vector/matrix pair. This process can proceed recursively, following a parse tree to produce a composite sentence meaning.

Chapter 3

Process Knowledge Extraction using Feature Engineering

Processes are complex events with many participating entities and inter-related sub-events. With the work in this chapter, we aim for macro-level role-based knowledge about processes. Our task is to find classes of entities that are likely to fill key roles within a process (*i.e.,* FRAME REPRESENTATION) namely, the *undergoer*, *enabler*, *result*, and *action*. Table 3.1 shows some examples of the target knowledge roles.

Process	**Undergoer**	**Enabler**	**Action**	**Result**
evaporation	liquid water	heat heat energy	changes convert	gas water vapor
weathering	rock solid material	weather heating	disintegration breaking down	smaller rocks smaller particles
photosynthesis	carbon dioxide CO2	solar energy light energy	convert transforms	energy food

Table 3.1: Examples of Target Knowledge Roles

Existing SRL systems extract semantic roles from a single sentence. In our case, we have several sentences describing a process and each of these sentences have similar entities filling similar semantic roles consistently. This allows us to design a joint inference method that can promote expectations of consistency amongst the extracted role fillers. This chapter presents a technique to extend the *within-sentence* inference in SRL to cross-sentence inference such that it encourages compatible role assignments across different sentence in a process. Section 3.1 briefly a simple cross-sentence inference technique. Section 3.2 presents an extension to cross-sentence inference with alignment variables. Section 3.3 presents the experimental results.

3.1 Simple Cross-Sentence Inference

Given a set of sentences about a process, we want to extract role fillers that are globally consistent i.e., we want role assignments that are compatible. Our approach is based on two observations: First, any given role is likely to have similar fillers for a particular process. For instance, the undergoers of the evaporation process are likely to be similar – they are usually liquids. Second, it is unlikely for fillers that are similar to have different roles for the same process. Using the same example as before, it is highly unlikely that one liquid *water* is an undergoer for evaporation, whereas another similar filler *ocean* is not. These role-specific selectional preferences vary for each process and can be learned if there are enough example role fillers for each process during training [46, 47]. Since, we wish to handle processes for which we have no training data, we approximate this by modeling whether two arguments should receive the same role given their similarity and their context similarity.

Figure 3.1 illustrates a formalization of our cross sentence inference approach using a factor graph. It includes one random variable for every candidate argument obtained from all the sentences for a given process: S_{ij} indicates the role label for argument j in sentence i. Each assignment to a argument S_{ij} is scored by a combination of the role classifier's score (factor ϕ_{role}), and its pairwise compatibility with the assignments to other arguments (factor ϕ_{align}). The factors ϕ_{sent} capture two basic within sentence constraints.

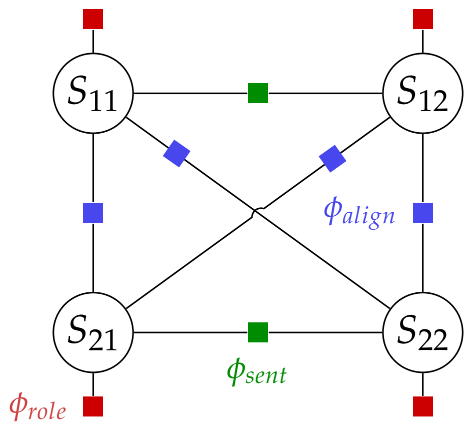

Figure 3.1: Factor graph representation of cross sentence inference. S_{11} and S_{12} denote role assignments for arguments a_{11} and a_{12} in one sentence, and S_{21} and S_{22} denote for arguments a_{21} and a_{22} in another. The ϕ_{role} factors score each role assignment to the arguments, and ϕ_{align} score the compatibility of the connected arguments. ϕ_{sent} encode sentence level constraints.

3.1.1 Role Classifier (ϕ_{role})

Given a set of process sentences, we build a role classifier to find the role fillers mentioned in them. Although existing SRL and frame semantic parsers do not directly produce the role information we need, we build on them by using their outputs for a process role classifier.

Specifically, we adapt EasySRL [30], a state-of-the-art SRL system to generate the candidate argument spans for each predicate (verbs) in the sentence. We use liblinear [48] to relabel these arguments and the predicates for our four roles, and an additional NONE role. The classifier is trained with a set of annotated examples (see Section 3.3). We use four sets of features to train the classifier:

i) **Lexical and Syntactic** – We use a small set of standard SRL features such as lexical and syntactic contexts of arguments (e.g., head word, its POS tag) and predicate-argument path features (e.g, dependency paths). We also add features that are specific to the nature of the process sentences. In particular, we encode syntactic relationships of arguments with respect to the process name mention in the sentence.

ii) **PropBank roles** – While they do not have a 1-to-1 correspondence with process roles, we use the EasySRL roles coupled with the specific predicate as a feature to provide useful evidence towards the process role.

iii) **Framenet Frames** – We use the frames evoked by the words in the sentence to allow better feature sharing among related processes. For instance, the contexts of undergoers in evaporation and condensation are likely to be similar as they are both state changes which evoke the same `Undergo_Change` frame in FrameNet.

iv) **Query patterns** – We use query patterns to find sentences that are likely to contain the target roles of interest. The query pattern that retrieved a sentence can help bias the classifier towards roles that are likely to be expressed in it.

3.1.2 Alignment Classifier (ϕ_{align})

We develop an alignment classifier to identify arguments that should receive similar role labels. One way to do this argument alignment is to use textual entailment. We used an approach that combined WordNet-based phrase similarity method, and word2vec vector similarity, where the vectors where learned from a general news domain. Entailment (or) argument similarity by itself is not enough to reliably figure out roles that should receive the same label. Moreover, the enabler, and the result roles are often long phrases whose text-based similarity is not reliable. Therefore, we build a classifier that uses many features that measure various aspects of compatibility of a pair of arguments, including the lexical and syntactic similarity of the arguments and the context in which they are embedded.

Fortunately, learning this classifier does not require any additional training data. The original data with annotated semantic role labels can be easily transformed to generate supervision for this classifier. For any given process, we consider all pairs of arguments in different sentences (i.e., (a_{ij}, a_{lm}) :

$$\arg\max_{\mathbf{z}} \sum_k \sum_{i,j} z_{ijk} \left(\lambda \underbrace{\phi_{role}(a_{ij},k)}_{\text{Role classifier score}} + (1-\lambda) \underbrace{\left[\Delta(a_{ij},k) - \nabla(a_{ij},k) \right]}_{\text{Global compatibility}} \right)$$

where compatibility with same roles is:

$$\Delta(a_{ij},k) = \frac{1}{\tilde{N}_k} \sum_{l,m} z_{lmk} \phi_{align}(a_{ij}, a_{lm})$$

and compatibility with other roles is:

$$\nabla(a_{ij},k) = \frac{2}{\tilde{N}_{k'}} \sum_{l,m} \sum_{n \neq k} z_{lmn} \underbrace{\phi_{align}(a_{ij}, a_{lm})}_{\text{Penalty when role n} \neq \text{k}}$$

subject to:

$$\sum_k z_{ijk} \leq 1 \qquad \forall\, a_{ij} \in \text{sentence}_i$$

$$\sum_j z_{ijk} \leq 1 \qquad \forall\, a_{ij} \in \text{sentence}_i, k \in \text{R}$$

$\tilde{N}_k$: Approximate number of arguments with role k

$\tilde{N}_{k'}$: Approximate number of arguments with role $n \neq k$

Figure 3.2: An Integer Linear Program formulation of the Cross-sentence Inference.

$i \neq l$) and label them as aligned if they are labeled with the same role, or unaligned otherwise.

3.1.3 Inference using ILP

We formulate inference as an Integer Linear Program shown in Figure 3.2. The goal is to optimize a combination of individual role assignment scores and their global compatibility, which is defined as the similarity of fillers for the same role minus the similarity of fillers of different roles.

The z_{ijk} variables are decision variables which denote role assignments to arguments. When z_{ijk} is set it denotes that argument j in sentence i (a_{ij})

has been assigned role k. The objective function uses three components to assign scores to an assignment.

1. Role Classifier Score $\phi_{role}(a_{ij},k)$ – The sentence-level role classifier's score for assigning role k to argument a_{ij}.

2. Within Role Compatibility $\Delta(a_{ij},k)$ – This is a measure of compatibility with other role k assignments. Specifically, this is an estimate of how well argument a_{ij} aligns with the other arguments that are also currently assigned role k. We normalize the sum of alignment scores using $(1/\tilde{N}_k)$, the total number of arguments in other sentences that can also receive the label k.

3. Across Role Incompatibility $\nabla(a_{ij},k)$ – This is a measure of how well a_{ij} aligns with the other arguments that are assigned a different role ($n \neq k$). For good assignments this quantity should be low and therefore we add this as a penalty to the objective. As with Δ, we use an approximation for normalization $(1/\tilde{N}_{k'})$, which is the product of other roles and the number of arguments in other sentences that can receive these roles. Because $\tilde{N}_{k'}$ is typically higher, we boost this score by 2 to balance against Δ.

Last, we use two sets of hard constraints to enforce the standard within-sentence constraints:

1. A single argument can receive only one label.

2. A sentence cannot have more than one argument with the same label, except for the NONE role.

We use an off-the-shelf solver in Gurobi (www.gurobi.com) to find an approximate solution the resulting optimization problem.

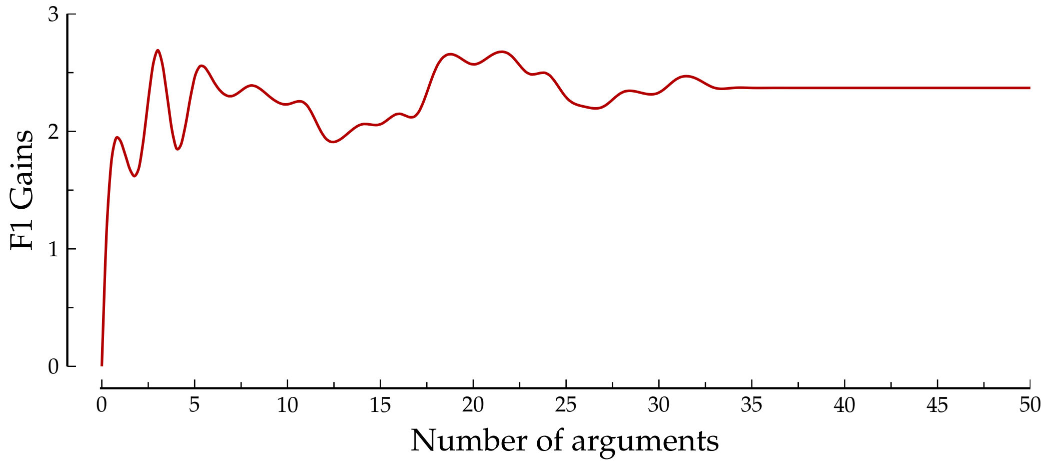

Figure 3.3: Cross sentence gains in F1 when varying the number of most similar arguments used to assess compatibilities.

3.2 Cross-Sentence Inference with Alignment Variables

We also studied the effect of varying the number of arguments that ILP uses to measure the compatibility of role assignments. Specifically, we allow inference to use just the top k alignments from the alignment classifier. Figure 3.3 shows the main trend. Using just the top similar argument already yields a 1 point gain in F1. Using more arguments tends to increase gains in general but with some fluctuations.

At some of the smaller argument counts we see a slightly larger gain (+0.3) compared to using all spans. This hints at benefits of a more flexible formulation that makes joint decisions on alignment and role label assignments.

Figure 3.4 illustrates a formalization of our cross sentence inference approach with alignment variables using a factor graph. It includes one random variable for every candidate argument obtained from all the sentences for a given process: S_{ij} indicates the role label for argument j in sentence i. Each assignment to a argument S_{ij} is scored by a combination of the role classifier's score (factor ϕ_{role}), and its pairwise compatibility with the assignments to other arguments (factor ϕ_{align}). Each ϕ_{align} factor has one additional random variable A_l to indicate whether a pair of candidate arguments align. The factors ϕ_{sent} capture two basic within sentence constraints.

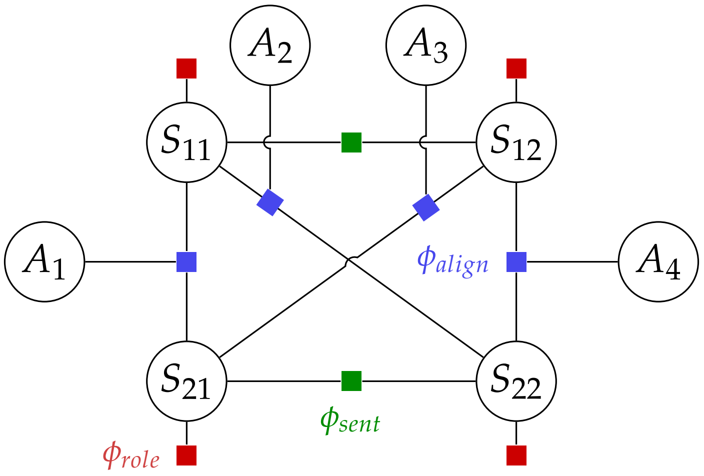

Figure 3.4: Factor graph representation of cross sentence inference with alignment variables. S_{11} and S_{12} denote role assignments for arguments a_{11} and a_{12} in one sentence, and S_{21} and S_{22} denote for arguments a_{21} and a_{22} in another. The ϕ_{role} factors score each role assignment to the arguments, and ϕ_{align} score the compatibility of the connected arguments. $A_1, \ldots, A_4$ indicate alignment assignments. ϕ_{sent} encode sentence level constraints.

We formulate inference as an Integer Linear Program shown in Figure 3.5. The goal is to optimize a combination of individual role assignment scores and their global compatibility, which is defined as the similarity of fillers for the same role minus the similarity of fillers of different roles.

The z_{ijk} variables are decision variables which denote role assignments to arguments. When z_{ijk} is set it denotes that argument j in sentence i (a_{ij}) has been assigned role k. The x_{ijlm} variables are decision variables which denote alignment between two arguments a_{ij} and a_{lm}. x_{ijlm} can take one of three values -1, 0 and 1. When x_{ijlm} is set to 1, it denotes that the argument a_{ij} aligns with the argument a_{lm}. When x_{ijlm} is set to -1, it denotes that the argument a_{ij} negatively aligns with the argument a_{lm} (dissimilarity). When x_{ijlm} is set to 0, it denotes that the alignment between the arguments a_{ij} and a_{lm} are ignored.

$$\arg\max_{\mathbf{z}} \sum_k \sum_{i,j} z_{ijk} \left(\lambda \underbrace{\phi_{role}(a_{ij},k)}_{\text{Role classifier score}} + (1-\lambda) \underbrace{\left[\Delta(a_{ij},k) - \nabla(a_{ij},k) \right]}_{\text{Global compatibility}} \right)$$

where compatibility with same roles is:

$$\Delta(a_{ij},k) = \frac{1}{\tilde{N}_k} \sum_{l,m} t_{ijlmk} \phi_{align}(a_{ij},a_{lm})$$

and compatibility with other roles is:

$$\nabla(a_{ij},k) = \frac{2}{\tilde{N}_{k'}} \sum_{l,m} \sum_{n \neq k} t_{ijlmn} \underbrace{\phi_{align}(a_{ij},a_{lm})}_{\text{Penalty when role } n \neq k}$$

subject to:

$$\sum_k z_{ijk} \leq 1 \qquad \forall\, a_{ij} \in \text{sentence}_i$$

$$\sum_j z_{ijk} \leq 1 \qquad \forall\, a_{ij} \in \text{sentence}_i, k \in \text{R}$$

$$t_{ijlmk} \leq z_{ijk}$$

$$t_{ijlmk} \geq -z_{ijk}$$

$$t_{ijlmk} \leq x_{ijlm} + (1 - z_{ijk})$$

$$t_{ijlmk} \geq x_{ijlm} - (1 - z_{ijk})$$

$\tilde{N}_k$: Approximate number of arguments with role k

$\tilde{N}_{k'}$: Approximate number of arguments with role $n \neq k$

Figure 3.5: An Integer Linear Program formulation of the Cross-sentence Inference with alignment variables.

As described in the previous section, the objective function uses three components to assign scores to an assignment.

1. Role Classifier Score $\phi_{role}(a_{ij},k)$
2. Within Role Compatibility $\Delta(a_{ij},k)$
3. Across Role Incompatibility $\nabla(a_{ij},k)$

Last, we use the following constraints to enforce the standard within-sentence constraints:

1. A single argument can receive only one label.
2. A sentence cannot have more than one argument with the same label, except for the NONE role.
3. Four additional constraints as described in Figure 3.5 to include alignment random variable x_{ijlm} in the formulation. [1]

3.3 Evaluation

Our goal is to generate knowledge about processes discussed in grade-level science exams. Since existing semantic resources such as FrameNet do not provide adequate coverage for these, we created a dataset of process sentences annotated with the four process roles: undergoer, enabler, action, and result.

This dataset consists of 1205 role fillers extracted from 537 sentences retrieved from the web. We first compiled the target processes from a list of process-oriented questions found in two collections: (i) New York Regents science exams [49], and (ii) `helpteaching.com`, a Web-based collection of practice questions. Then, we identified 127 process questions from which we obtained a set of 180 unique target processes. For each target process, we queried the web using Google to find definition-style sentences, which describe the target process.

Table 3.2 shows some examples of the 14 query patterns that we used to find process descriptions. Because these patterns are not process-specific, they work for unseen processes as well.

To find role fillers from these sentences, we first processed each sentence

[1]Since we cannot multiply two random variables (z_{ijk} and x_{ijlm}) and have Gurobi optimize it, we use a single random variable t_{ijlmk} in the formulation and add the four constraints on t_{ijlmk} which use z_{ijk} and x_{ijlm} to replicate the multiplication.

Query Patterns
⟨name⟩ is the process of ⟨x⟩
⟨name⟩ is the process by which ⟨x⟩
⟨name⟩ {occurs when} ⟨x⟩
⟨name⟩ { helps to | causes } ⟨x⟩

Table 3.2: Example query patterns used to find process description sentences.

Role	**No. of instance**
Undergoer	77
Enabler	154
Action	315
Result	194
NONE	465

Table 3.3: Role distribution

using EasySRL [30] to generate candidate arguments. Some of the query patterns can be used to generate additional arguments. For example, in the pattern "⟨name⟩ is the process of ⟨x⟩" if ⟨x⟩ is a noun then it is likely to be an undergoer, and thus can be a good candidate[2]. Then two annotators annotated the candidate arguments with the target roles if one were applicable and marked them as NONE otherwise. Disagreements were resolved by a third annotator. The annotations spanned a random subset of 54 target processes.

The role label distribution is shown in Table 3.3.

We conducted five fold cross validation experiments to test role extraction. To ensure that we are testing the generalization of the approach to unseen processes, we generated the folds such that the processes in the test fold were unseen during training. We compared the basic role classifier described in Section 3.1.1, the *within sentence* and the *cross sentence* inference models. We tune the ILP parameter λ for cross sentence inference based on a coarse-grained sweep on the training folds.

[2]These patterns are not unambiguous and are not adequate by themselves for extracting the roles. We use them as features.

We also compared with a simple baseline that learned a mapping from PropBank roles produced by EasySRL system to the process roles by using the roles and the verb as features. We also add the FrameNet frames invoked by the lexical unit in the sentence. Note this is essentially a subset of the features we use in our role classifier. As a second baseline, we compare with a (nearly) out-of-the-box application of SEMAFOR [33], a FrameNet based frame-semantic parser. We modified SEMAFOR to override the frame identification step since the process frame information is already associated with the test sentences.

Table 3.4 compares the performance of the different methods. The learned role mapping of shallow semantic roles performs better than SEMAFOR but worse than the simple role classifier. SEMAFOR uses a large set of features which help it scale for a diverse set of frames in FrameNet. However, many of these many not be well suited for the process sentences in our relatively smaller dataset. Therefore, we use our custom role classifier as a strong baseline to demonstrate within and cross sentence gains.

Method	Prec.	Rec.	F1
Role mapping	56.62	59.60	58.07
SEMAFOR	40.72	50.54	45.10
Role class. (ϕ_{role})	78.48	**78.62**	78.55
+ within sent.	86.25	73.91	79.60
+ cross sent.	**89.84**	75.36	**81.97**††
+ cross sent. w/ align	86.68	74.27	80.00

Table 3.4: Process role inference performance. †† indicates significant improvement over Role + within sentence system.

The role classifier baseline, which operates at a sentence level without any consistency constraints performs well with roughly the same precision and recall values. Enforcing sentence-level consistency through joint inference shown as (+within sent.) mainly serves to increase precision (by nearly 8 points), while loosing recall in the trade-off (by about 4.7 points) and yields an overall gain in F1 by 1.05 points. Cross sentence inference provides additional gains beyond within sentence inference by another 2.38 points in F1

[3]. Cross sentence inference with alignment variables provides gains beyond within sentence inference, but it does not perform as well as cross sentence inference without alignment variables. Though the inclusion of alignment variables allows the model to be flexible but the ILP optimizer fails to select best aligning spans the during optimization process.

Figure 3.6 shows the precision/recall plots for inference using within, cross sentence and cross sentence with alignment variables as compared to the basic role classifier. The inference models trade recall for gains in precision. Cross sentence yields higher precision at most recall levels, for a smaller overall loss in recall compared to within sentence (1.6 versus 4.9). And, performance of cross sentence with alignment model lies in between within sentence and cross sentence versions.

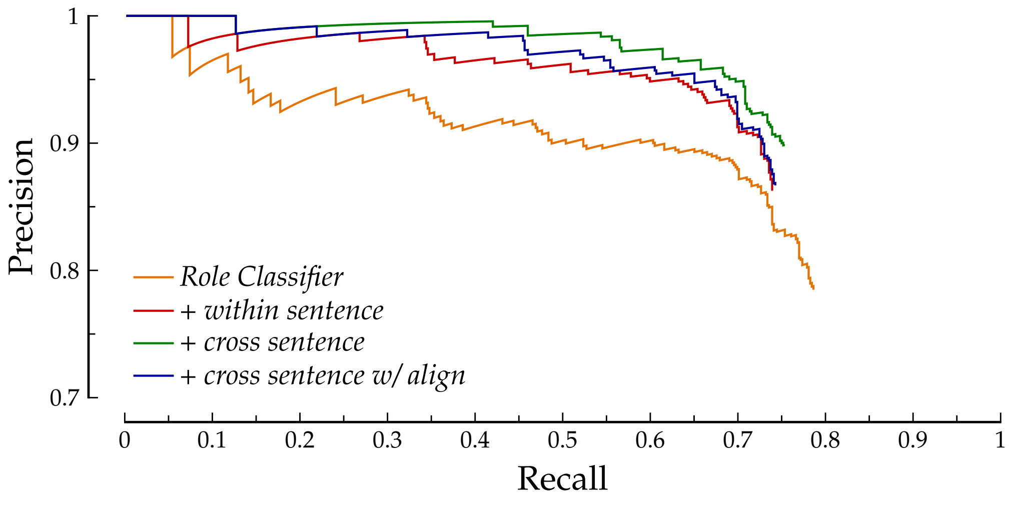

Figure 3.6: Precision/Recall trade-offs for process role inference. y-axis is truncated at 0.7 to better visualize the differences.

Simple role-based knowledge is essential for recognizing and reasoning about situations involving processes. In this work we developed a method for automatically acquiring such role-based knowledge for new processes. The main idea is to enforce compatibility among roles extracted from sentences belonging to a single process. We also find that using the same supervised training data, we learn an alignment classifier that can predict which

[3]The single parameter in ILP turned out to be stable across the folds and obtained this best value at $\lambda = 0.8$.

arguments should receive the same labels. Using outputs from a custom-built feature engineered role classifier, and the alignment classifier we formulated an Integer Linear Program to extract globally consistent role labels. Our evaluations on a process dataset shows that this approach helps improve extraction accuracy, showing the potential for generalizing extraction to new unseen processes.

Chapter 4

Process Knowledge Extraction using LSTM

As described in the previous chapter, processes are complex events with many participating entities and inter-related sub-events. And, with the work in this chapter, we aim for macro-level role-based knowledge about processes without feature engineering.

We aim to build a role classifier using sequence-to-sequence models without any feature engineering. As discussed in the introduction and background section of this thesis, neural networks can perform the task of jointly learning feature representations and feature weights. A well-studied solution for a neural network to process variable length input is the recurrent neural network (RNN). LSTM, a variant of RNN with memory cell for long term dependencies has shown great success in diverse NLP tasks such as speech recognition, machine translation, and language modeling.

We have long known the importance of long-term dependencies in language. But to keep sparsity in check, traditional models had to rely on independence assumptions. With LSTMs we replace the explicit independence assumptions in text processing with structural constraints on memory. This allows LSTMs to be more flexible in efficiently handling long-term dependencies. Role classification is a task in which modeling long-term dependencies help in developing accurate systems. In this chapter, we describe a LSTM based architecture for role classification. Cross-sentence inference is then performed using the ILP based method described in the previous chapter.

4.1 Role Classification using LSTM

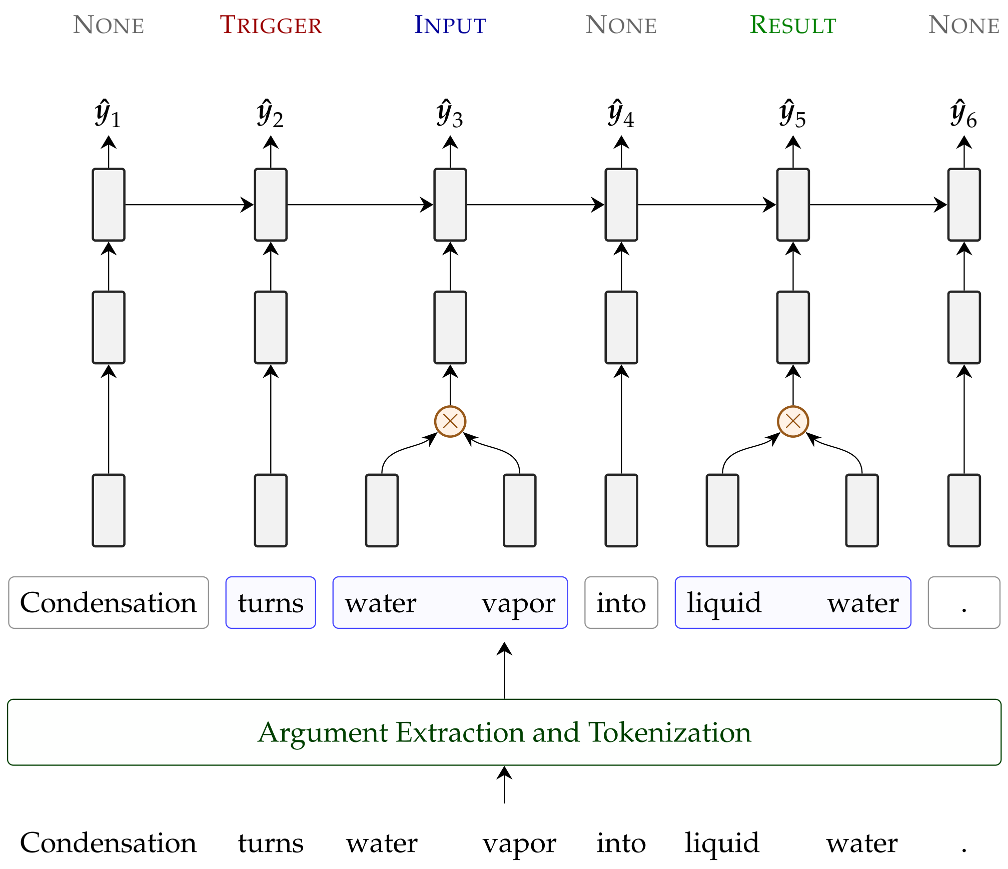

Figure 4.1: LSTM based Role Classifier Pipeline

Figure 4.1 illustrates the LSTM based role classifier pipeline. We use EasySRL and query patterns to generate candidate argument spans from raw text as described in the previous chapter. Once we have the candidate argument spans, we discard overlapping spans[1]. The tokenizer considers the extracted candidate argument spans and keeps the phrasal spans intact while tokenizing (*e.g.,* phrasal tokens in blue boxes in the figure). LSTM needs vectorial representation of input in order to perform classification. We use GloVe vectors [50] to represent each word. In case of phrasal tokens, we multiply the word vector of each of the words in the phrase to get the phrase vectors. These are then passed through the hidden layer of LSTM to train (and predict) role labels. And, as shown in the figure, every token has one of 5 possible gold labels (None, Input, Result, Enabler, Trigger).

Back-propagation is then used to learn the weights at the hidden layers. We applied RMSprop[51] for optimization, which is more suitable for training RNNs than naïve stochastic gradient descent, and less sensitive to hyperparameters compared with momentum methods.

We use the same ILP based formulation described in the previous chapter for cross-sentence inference. But instead of using feature engineered role classifier, we use the LSTM classifier described above to get role probabilities ϕ_{role}. For alignment values ϕ_{align}, we use the cosine similarity between phrasal embeddings of the phrases that are considered for alignment.

4.2 Evaluation

We used the same dataset that is described in the previous chapter. Since LSTMs consume sequential data, overlapping argument spans—after argument extraction—cannot be used directly in the architecture described above. Hence we consider only the smaller argument spans in cases where we have overlapping spans and ignore the bigger ones. This results in a minor change in dataset size. Here we have 1021 role fillers as compared to 1205 considered for the model in the previous chapter.

We conducted five fold cross validation experiments to test role extraction.

[1]We consider only the smaller argument spans in cases where we have overlapping spans and ignore the bigger ones.

To ensure that we are testing the generalization of the approach to unseen processes, we generated the folds such that the processes in the test fold were unseen during training. We conducted experiments with LSTMs of different depths. Table 4.1 compares the performance of these models where the depth of model is included within parenthesis. We use ILP based optimization described in the previous chapter for cross-sentence inference over LSTM outputs. From the results, it is evident the depth helps in increasing the accuracy of the LSTM classifier. As the depth increases, we don't see any significant gains in LSTM classifier, but when coupled with cross sentence inference, we see significant gains. This is because, with higher depths, the deeper hidden state vectors summarize a larger portion of the sentence and hence tends to give better probability distribution over roles. This enables ILP to easily make minor changes using similarity values and constraints. This is evident from the results.

Method	Prec.	Rec.	F1
LSTM (1)	86.17	66.62	75.14
LSTM (1) w/ cross sent.	81.63	68.97	74.77
LSTM (2)	86.33	67.5	75.76
LSTM (2) w/ cross sent.	81.13	74.14	77.48
LSTM (3)	**88.8**	68.52	77.36
LSTM (3) w/ cross sent.	88.46	**79.31**	**83.64**

Table 4.1: Performance of LSTM models with varying number of hidden layers.

Table 4.2 compares the best feature engineered model described in the previous chapter with the best LSTM based model. LSTM classifier has higher precision than the LIBLINEAR classifier (by nearly 10 points). But it has a comparatively poor recall. With cross-sentence inference, LSTM outperforms LIBLINEAR model (by 1.67 points in F1).

When training a neural network, finding the optimal parameter can provide enormous gains in performance. However, due to time constraints, our LSTM has not been optimized. The parameter settings used for the experiment are listed in Table 4.3. Finding the optimal parameter settings can provide a much bigger boost to the system performance.

Method	Prec.	Rec.	F1
LIBLINEAR role class.†	78.48	78.62	78.55
LIBLINEAR role class.† w/ cross sent.	**89.84**	75.36	81.97
LSTM (3)	88.8	68.52	77.36
LSTM (3) w/ cross sent.	88.46	**79.31**	**83.64**

Table 4.2: Process role inference performance. † indicates the feature engineered role classifier described in the previous chapter.

Parameter	**Value**
Word embedding size	100
Learning rate	0.001
Dropout rate	0.5
Number of epochs	30

Table 4.3: LSTM Parameters.

In this work we developed a method for automatically acquiring role-based knowledge for new processes without any feature engineering. The main idea here is to use deep LSTM models to perform role classification for any new process. LSTMs can effectively utilize the shallow semantic information based on co-occurrence statistics embedded in the word vectors to make decisions on semantic role labels. This model coupled with ILP based cross-sentence inference mechanism shows improved extraction accuracy.

Chapter 5

Modeling Processes as Operators

Processes are complex events with several participating entities. It is not possible to have a scalable mechanism that includes such complexity in a fixed frame representation. For example, the 'rotation' process cannot be represented in FRAME REPRESENTATION (with *Input*, *Output*, *Trigger* and *Enabler* roles) because a representation for rotation is incomplete without the information about the axis of rotation. One alternative is to have different fixed frames for different processes. Such representation methods can easily handle this issue, but they are not scalable.

In this chapter, we explore the possibility of using a matrix for process representation. We consider the simple version of the process representation problem where the process matrix consumes *input* and emits an *output*. In future, this can be scaled for multiple roles by designing a mechanism based on matrix-vector operations over role specific vectors.

Section 5.1 describes a method based on WSABIE model [52] for learning process operators. Section 5.2 describes the methods considered for data collection and the difficulty involved in the collection process.

5.1 Model

Our goal is to model processes as operators (MATRIX REPRESENTATION) in order to predict the output of a process given its input. Example: given water *(input)* and evaporation *(process)*, predict water vapor *(output)*.

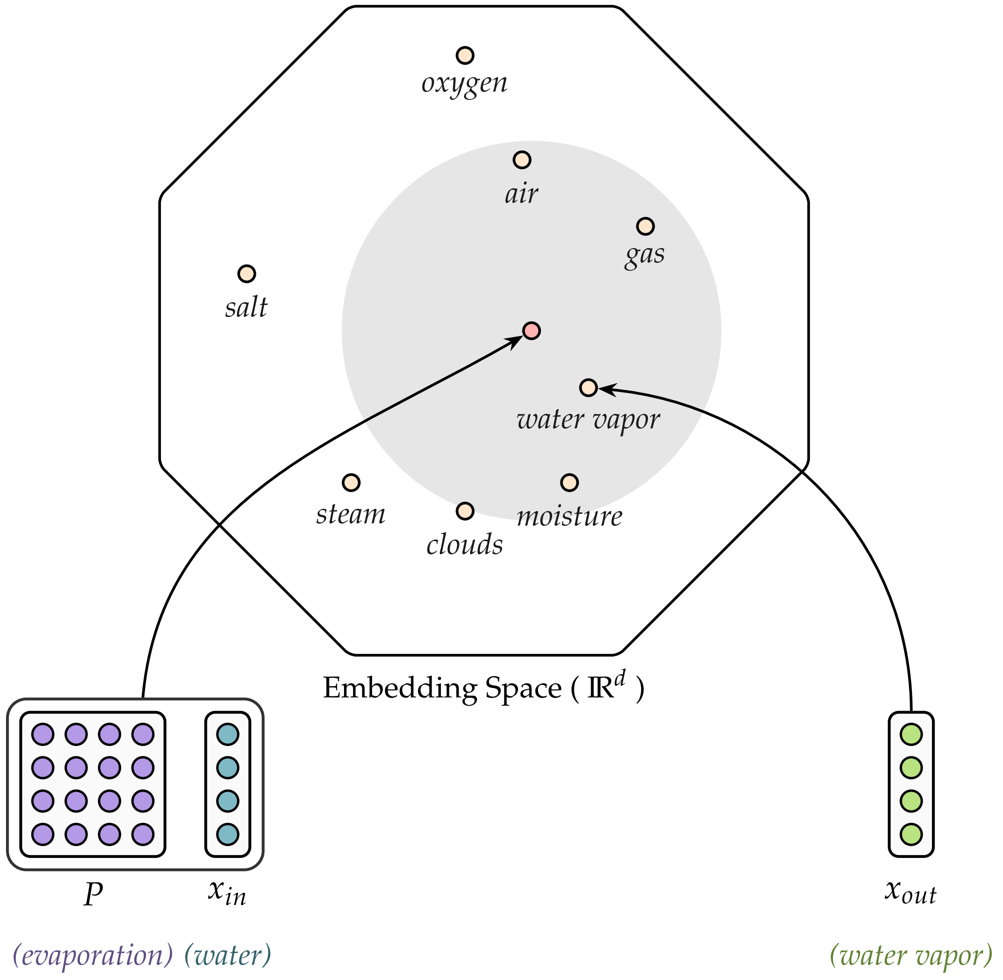

Figure 5.1: Process Operator Modeling

We begin by associating each word w in our vocabulary with a vector representation $x_w \in \mathbb{R}^d$. These vectors are stored as the columns of a $d \times V$ dimensional word embedding matrix W_e, where V is the size of the vocabulary. We propose to learn a mapping matrix—we call this the process matrix— P that takes input word vector x_{in} and maps it onto a point in embedding space that is very close to output word vector x_{out}. Figure 5.1 illustrates this using the example of water evaporating to form water vapor.

Our goal is to rank the possible outputs of a given input such that the more probable outputs are ranked higher than the others. We consider the following model to score the input-output compatibility:

$$s(in,out) = (x_{out})^T P x_{in}$$

The possible outputs are ranked according to the magnitude of $s(in,out)$, which calculates the dot product between the vector x_{out} and the projected input vector Px_{in}. Our goal is to learn the model parameter $P \in \mathbb{R}^{d\times d}$ such that it projects x_{in} very close to probable outputs and far away from improbable outputs. To learn the parameter P, we model our objective function following Weston et al. [52], using a weighted approximate rank pairwise (WARP) loss, learned with stochastic gradient descent (SGD).

The training objective function minimizes:

$$\sum_{x}\sum_{\overline{y}} L\left(rank_y(x)\right) \max\left(0, \gamma + s\left(x,y\right) - s\left(x,\overline{y}\right)\right)$$

where x, y are the training inputs and their corresponding correct outputs, $\overline{y}$ are negative outputs, and γ is the margin. Here, $rank_y(x)$ is the rank of the positive output y relative to all the negative outputs:

$$rank_y(x) = \sum_{\overline{y}} I\left(s(x,y) \leq \gamma + s(x,\overline{y})\right)$$

where I is the indicator function, and $L(\cdot)$ converts the rank into a weight for the loss.

$$L(\eta) = \sum_{j=1}^{\eta} 1/j$$

One can define different choices of $L(\cdot)$ with different minimizers. The chosen $L(\cdot)$ optimizes the top of the ranked list [53]. To train with such an objective, stochastic gradient is employed. For speed, the computation of $rank_y(x)$ is then replaced with a sampled approximation: sample N items $\overline{y}$ until a violation is found, i.e. $max(0, \gamma + s(x, \overline{y}) - s(x, y))) > 0$ and then approximate the rank with

$$rank_y(x) \approx \left\lfloor \frac{Y-1}{N} \right\rfloor$$

where Y is the number of possible outputs.

5.2 Experiments & Challenges

Our goal is to model processes as operators. This needs a large number of varying *input-output* pairs for each of the processes. There is no large scale training data that can be readily used for this task. We tried the following methods to collect a large number of varying, process specific *input-output* pairs.

- *Automatic pattern-based search*: We built a system to retrieve sentences that follow pre-defined patterns from the web. Methods described in Chapter 3 were then used to extract *input-output* pairs.

- *Bootstrapped data collection*: We built a system to retrieve sentences using a bootstrapped mechanism to collect several *outputs* for a given *input*. The output search is done based on pattern-based technique where input is included within the pattern.

- *Manual search*: 2 students including me spent $\sim$ 6 hours manually searching for sentences with good *input-output* pairs.

- *Crowdsourcing*: We ran few experiments on Amazon's Mechanical Turk (AMT) to collect *input-output* pairs and associated sentences.

We found the task of collecting *input-output* pairs to be challenging because, the number of possible distinct *input-output* pairs for the processes in consideration is limited. And, these are the same ones that are repeated in different web resources (Most of the sentences that describe/mention evaporation have a small set of lexicons representing liquid or water as input and gas or vapor as output). If we expand our definition of output to include conclusive events of a process (this is usually not a single entity or a direct result of the process *e.g.,* Evaporation of water body makes its water salty.), we can expect more variety in the kinds of spans we can collect. But, this makes the task challenging. We tried the above listed methods to collect spans and with these methods we collected ~ 40 *input-output* pairs for the process evaporation and $\sim 10 - 15$ *input-output* pairs for three other processes.

This data is not sufficient to perform conclusive experiments. So, we used it to do a preliminary test of our idea without using it to arrive at any significant conclusions. We used the 37 pairs collected for the evaporation process to jointly learn word embedding and process matrix using the model described above. We evaluated the quality of the learnt word embeddings and operator manually by inspecting the outputs returned by the model for specific inputs. The leant process matrix behaved as expected, but this could be because we are using a powerful model on small dataset and hence the model could be overfitting. Further exploration of this idea needs more data. This preliminary work can help us in deciding the next steps for further exploration.

Chapter 6

Conclusion

6.1 Conclusions

In this thesis, we have demonstrated the following:

- We show that a small set of semantic roles can be used to build an effective representation (FRAME REPRESENTATION) for recognizing and reasoning about situations involving processes.

- We present a feature engineered system for role classification and alignment classification. Using outputs from the role classifier, and the alignment classifier, we formulated an Integer Linear Program to extract globally consistent role labels (cross-sentence inference). Our evaluations on a process dataset shows that cross-sentence inference helps improve extraction accuracy, showing the potential for generalizing extraction to new unseen processes.

- We then present a method for automatically acquiring role-based knowledge for new processes without any feature engineering. We demonstrate improved extraction accuracy from deep LSTM models in conjunction with cross-sentence inference.

- We also present our preliminary work on using MATRIX REPRESENTATION for modeling processes.

6.2 Future Work

One can explore the following research directions from this thesis, which are enumerated below:

- One could jointly model role classification and alignment classification using attention mechanism over LSTMs. This is an end-to-end approach and these kinds of end-to-end approaches have been successful recently in machine translation.

- Fixed set of roles can only be used to represent a small subset of processes. In order to scale one needs to design alternate scalable representations. We explored one possibility using MATRIX REPRESENTATION. One could explore that further.

  Another possibility is to use clustering to discover the roles in sentences that describe a process.

Bibliography

[1] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10): 2451–2471, 2000.

[2] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[4] Samuel Louvan, Chetan Naik, Veronica Lynn, Ankit Arun, Niranjan Balasubramanian, and Peter Clark. Semantic role labeling for process recognition questions. In *2015 Scientific Knowledge Capture Workshop*, 2015.

[5] Samuel Louvan, Chetan Naik, Sadhana Kumaravel, Heeyoung Kwon, Niranjan Balasubramanian, and Peter Clark. Cross sentence inference for process knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[6] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

[7] Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21, 2007.

[8] Michael Kaisser and Bonnie Webber. Question answering based on semantic roles. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 41–48. Association for Computational Linguistics, 2007.

[9] Mark Sammons, VG Vinod Vydiswaran, Tim Vieira, Nikhil Johri, Ming-Wei Chang, Dan Goldwasser, Vivek Srikumar, Gourab Kundu, Yuancheng Tu, Kevin Small, et al. Relation alignment for textual entailment recognition. In *Text Analysis Conference (TAC)*, 2009.

[10] Dekai Wu and Pascale Fung. Semantic roles for smt: a hybrid two-pass model. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 13–16. Association for Computational Linguistics, 2009.

[11] Ding Liu and Daniel Gildea. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 716–724. Association for Computational Linguistics, 2010.

[12] Qin Gao and Stephan Vogel. Corpus expansion for statistical machine translation with semantic role label substitution rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 294–298. Association for Computational Linguistics, 2011.

[13] Roberto Basili, Diego De Cao, Danilo Croce, Bonaventura Coppola, and Alessandro Moschitti. Cross-language frame semantics transfer in bilingual corpora. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 332–345. Springer, 2009.

[14] Lonneke Van der Plas, James Henderson, and Paola Merlo. Domain adaptation with artificial data for semantic parsing of speech. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 125–128. Association for Computational Linguistics, 2009.

[15] Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. URL `http://www.deeplearningbook.org`.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[17] Yoshua Bengio, Olivier Delalleau, and Nicolas L Roux. The curse of highly variable functions for local kernel machines. In *Advances in neural information processing systems*, pages 107–114, 2005.

[18] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[19] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5 (3):1, 1988.

[21] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[24] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, page 91, 1991.

[25] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[26] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[27] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[28] Luiz Augusto Pizzato and Diego Mollá. Indexing on semantic roles for question answering. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pages 74–81. Association for Computational Linguistics, 2008.

[29] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP*, 2014.

[30] Mike Lewis, Luheng He, and Luke Zettlemoyer. Joint a* ccg parsing and semantic role labelling. In *Empirical Methods in Natural Language Processing*, 2015.

[31] Anders Björkelund, Love Hafdell, and Pierre Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics, 2009.

[32] Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1220355.1220552. URL `http://dx.doi.org/10.3115/1220355.1220552`.

[33] Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. Probabilistic frame-semantic parsing. In *Proc. of NAACL-HLT*, 2010.

[34] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[35] Robert S Swier and Suzanne Stevenson. Unsupervised semantic role labelling. In *EMNLP*, 2004.

[36] Joel Lang and Mirella Lapata. Unsupervised semantic role induction with graph partitioning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1320–1331, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D11-1122`.

[37] Hagen Fürstenau and Mirella Lapata. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 11–20, Singapore, 2009.

[38] Hagen Fürstenau and Mirella Lapata. Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, 38(1):135–171, 2012.

[39] Joel Lang and Mirella Lapata. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California, June 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N10-1137`.

[40] Ivan Titov Alexandre Klementiev. Semi-supervised semantic role labeling: Approaching from an unsupervised perspective. In *Proceedings of the COLING Conference.*, 2012.

[41] Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime G. Carbonell, Noah A. Smith, and Chris Dyer. Frame-semantic role labeling with heterogeneous annotations. In *ACL*, 2015.

[42] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[43] Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015.

[44] Swantje Tönnis and Nadine Theiler. Nouns are vectors, adjectives are matrices. 2012.

[45] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

[46] Beñat Zapirain, Eneko Agirre, and Lluís Màrquez i Villodre. Generalizing over lexical features: Selectional preferences for semantic role classification. In *ACL*, 2009.

[47] Beñat Zapirain, Eneko Agirre, Lluís Màrquez i Villodre, and Mihai Surdeanu. Selectional preferences for semantic role classification. *Computational Linguistics*, 39:631–663, 2013.

[48] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[49] Peter Clark. Elementary school science and math tests as a driver for ai: Take the aristo challenge. *to appear*, 2015.

[50] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

[51] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[52] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. 2011.

[53] Nicolas Usunier, David Buffoni, and Patrick Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th annual international conference on machine learning*, pages 1057–1064. ACM, 2009.